In [ ]:
```
from sklearn.datasets import load_iris
import numpy as np
import pandas as pd
```

In [ ]:
```
iris = load_iris()
df = pd.DataFrame(iris.data, columns=iris.feature_names)
df
```

Out[ ]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
...	...	...	...	...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

150 rows × 4 columns

In [ ]:
```
df['Target'] = iris.target
df
```

Out[ ]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	Target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	2
146	6.3	2.5	5.0	1.9	2
147	6.5	3.0	5.2	2.0	2
148	6.2	3.4	5.4	2.3	2
149	5.9	3.0	5.1	1.8	2

150 rows × 5 columns

In [ ]:
```
X = iris.data
y = iris.target
```

In [ ]:
```
df.describe()
```

Out[ ]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	Target
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333	1.000000
std	0.828066	0.435866	1.765298	0.762238	0.819232
min	4.300000	2.000000	1.000000	0.100000	0.000000
25%	5.100000	2.800000	1.600000	0.300000	0.000000
50%	5.800000	3.000000	4.350000	1.300000	1.000000
75%	6.400000	3.300000	5.100000	1.800000	2.000000
max	7.900000	4.400000	6.900000	2.500000	2.000000

In [ ]:
```
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
```

In [ ]:
```
X_train, X_test, y_train, y_test=train_test_split(X, y, test_size=0.2)
```

In [ ]:
```
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(120, 4)
(30, 4)
(120,)
(30,)
```

Gaussian Naive Bayes

In [ ]:
```
gnb = GaussianNB()
gnb.fit(X_train, y_train)
```

Out[ ]:
```
GaussianNB()
```

In [ ]:
```
y_pred = gnb.predict(X_test)
y_pred
```

Out[ ]:
```
array([1, 1, 0, 1, 2, 1, 1, 2, 1, 0, 2, 2, 1, 0, 2, 0, 1, 2, 1, 1, 0, 1,
       2, 0, 2, 0, 1, 2, 1, 2])
```

In [ ]:
```
print("Accuracy :", str(metrics.accuracy_score(y_test, y_pred)))
```

```
Accuracy : 1.0
```

In [ ]:
```
print(gnb.predict(np.array([4.5, 2.8, 2.1, 0.2]).reshape(1,-1)))
print(gnb.predict(np.array([5.1, 3.5, 3.4, 1.6]).reshape(1,-1)))
print(gnb.predict(np.array([4.8, 2.9, 5.4, 2.4]).reshape(1,-1)))
```

```
[0]
[1]
[2]
```

KNN Classifier

In [ ]:
```
knn = KNeighborsClassifier(n_neighbors=7)
knn.fit(X_train, y_train)
```

Out[ ]:
```
KNeighborsClassifier(n_neighbors=7)
```

In [ ]:
```
y_pred_knn = knn.predict(X_test)
y_pred_knn
```

Out[ ]:
```
array([2, 1, 0, 1, 2, 1, 1, 2, 1, 0, 2, 2, 1, 0, 2, 0, 1, 2, 1, 1, 0, 1,
       2, 0, 2, 0, 1, 2, 1, 2])
```

In [ ]:
```
print("Accuracy :", str(metrics.accuracy_score(y_test, y_pred_knn)))
```

```
Accuracy : 0.9666666666666667
```

In [ ]:
```
print(knn.predict(np.array([4.5, 2.8, 2.1, 0.2]).reshape(1,-1)))
print(knn.predict(np.array([5.1, 3.5, 3.4, 1.6]).reshape(1,-1)))
print(knn.predict(np.array([4.8, 2.9, 5.4, 2.4]).reshape(1,-1)))
```

```
[0]
[1]
[2]
```